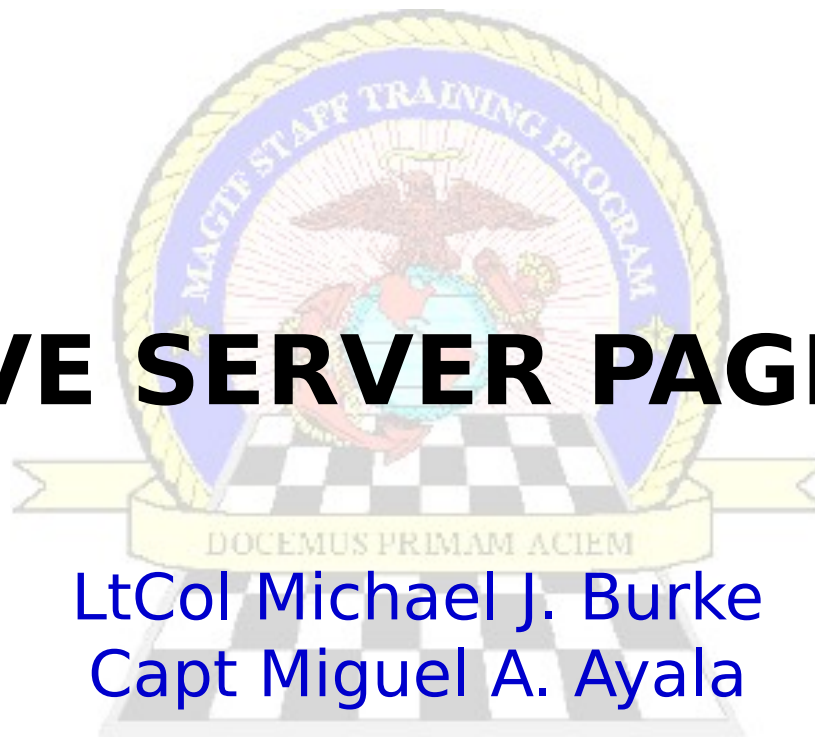


ASP 3.0



MSTP

ACTIVE SERVER PAGES 3.0



LtCol Michael J. Burke
Capt Miguel A. Ayala

"Training The First To Fight"

04/10/04

PURPOSE



MSTP

- To provide a basic overview of Active Server Pages.
 - What is it?
 - What could you do with it?
- To provide a beginners level of proficiency in the development of ASP applications.
- By the end of this class you should have the basic knowledge to develop an ASP application.

ASSUMPTIONS



MSTP

- Familiarity with HTML.
- At least a little exposure to a scripting language or programming language like VB, VBScript, Java or JScript.



WHAT DO WE NEED?

MSTP

- Obviously, you will need a web server.
 - Internet Information Server (IIS)
 - Personal Web Server (PWS)
- Browser
 - Internet Explorer
 - Netscape
- Editor
 - Notepad or any other text editor
 - Microsoft FrontPage

CLASS SCHEDULE



MSTP

- **HTML Basics**
 - **Lab 1: Text Formatting**
 - **Lab 2: Tables**
 - **Lab 3: Forms**
- **Getting Started**
 - **Lab 4: Your First ASP Code**
- **ASP Overview**
- **Programming and Scripting in VBScript**
- **Processing user input request (Request Object)**
 - **Lab 5: Using a form to gather user input and displaying results**
- **Session Object**
 - **Lab 6: Use of Session Variables**
- **Responding to the user (Response Object)**
 - **Lab 7: Responding to the user**
- **Database Overview**
- **Introduction to Structured Query Language (SQL)**
- **Accessing a Database**
 - **Lab 8: Database Connection**
- **Advance Topics**
- **Putting it all together**
- **Lab 9: Alpha Roster (Final Product)**

04/10/04

HTML



MSTP

HTML BASICS

04/10/04

HTML BASICS



MSTP

- What is HTML?
 - HTML is a document-layout and hyperlink-specification language.
 - It defines the syntax and placement of special, embedded directions that aren't displayed by the browser, but tell it how to display the contents of the document, including text and images.
 - The language also tells you how to make a document interactive through special hypertext links, which connect your document with other documents, on either your computer or someone else's, as well as with other Internet resources.



HTML BASICS

MSTP

- Basic Structure of an HTML Page

```
<HTML> ← Opening tag
  <HEAD>
    <TITLE>Barebones HTML Document</TITLE>
  </HEAD>
<BODY>
  This illustrates, in a very <i>simple</i> way,
  the basic structure of an HTML document.
</BODY>
</HTML> ← Closing tag
```


HTML BASICS



MSTP

- Tags
 - Marks a portion of the text for special treatment by the browser.
 - Every HTML Tag consists of a tag name, sometimes followed by an optional list of tag attributes, all placed between opening and closing brackets (< >).
 - Not case sensitive
 - <Head>...</Head>
 - <HEAD>...</HEAD>
 - <HeaD>...<HeaD>



HTML BASICS

MSTP

Text Formatting

- Heading Styles <H>...</H>
 - Six levels of heading:
 - <H1> through <H6>.
 - Number signifies the position of the heading hierarchy (smaller is higher in the hierarchy).

<HTML>

<HEAD>

<TITLE>HTML Document Test Headings</TITLE>

</HEAD>

<BODY>

This illustrates the different types of headings of an HTML document.

<H1> Heading 1 </H1>

<H2> Heading 2 </H2>

<H3> Heading 3 </H3>

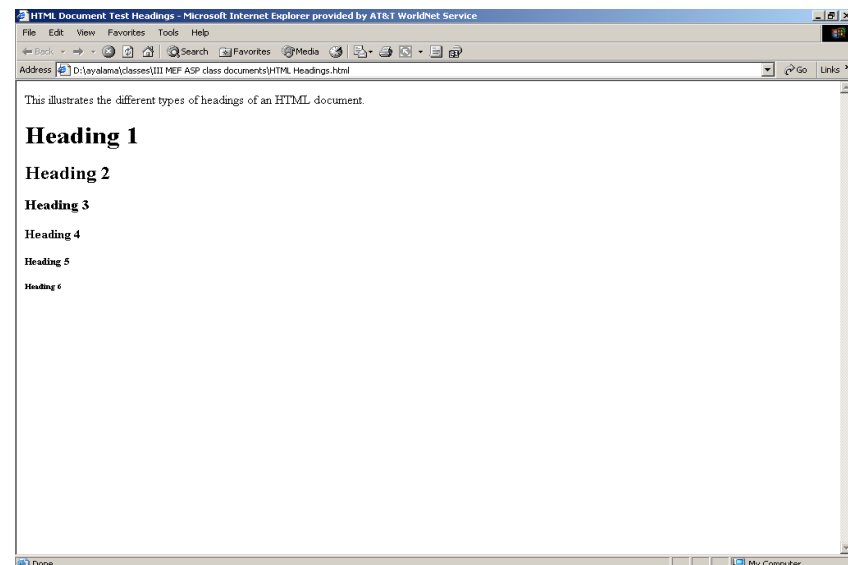
<H4> Heading 4 </H4>

<H5> Heading 5 </H5>

<H6> Heading 6 </H6>

</BODY>

</HTML>



04/10/04



HTML BASICS

MSTP

Text Formatting

- Paragraph `<P>...</P>`
 - While your browser will automatically adjust the end of the line to suit the window size of the browser in which it is being displayed, the `<P>` tag automatically forces a carriage return and forces another line (a new paragraph).
 - Can contain child tags, such as text formatting

```
<HTML>  
<HEAD>
```

```
  <TITLE>HTML Document Paragraph Headings</TITLE>
```

```
</HEAD>  
<BODY>
```

```
  This illustrates the paragraph tag of an HTML document.
```

```
    <p> paragraph 1 </p>  
    <p> paragraph 2 </p>  
    <p> paragraph 3 </p>
```

```
</BODY>  
</HTML>
```

04/10/04

HTML BASICS



MSTP

Text Formatting

- Break

 - Interrupts the normal line filling and word wrapping of paragraphs.
 - It has no ending tag.
 - Simply marks the point in the flow where a new line should begin.

```
<HTML>  
<HEAD>
```

```
<TITLE>HTML Document Fonts</TITLE>
```

```
</HEAD>  
<BODY>
```

This illustrates the Font tag of an HTML document.

```
<FONT SIZE = 8> <p> <b> paragraph 1 in bold font and size 8 </b> </p> </FONT>
```

```
<FONT SIZE = 5 > <p> <i> paragraph 2 in italics font and size 5 </i> </p></FONT>
```

After a long sentence you can simply add a break to specify
 where the next line should begin.

```
</BODY>  
</HTML>
```




HTML BASICS

MSTP

Text Formatting

- Font `...`
 - Used to modify the display of the characters.
 - Bold `...`
 - The `` tag explicitly boldfaces a character or segment of text that is enclosed between it and its corresponding `` end tag.
 - Italic `<i>...</i>`
 - The `<i>` tag renders the enclosed text between it and `</i>` end tag into italic.

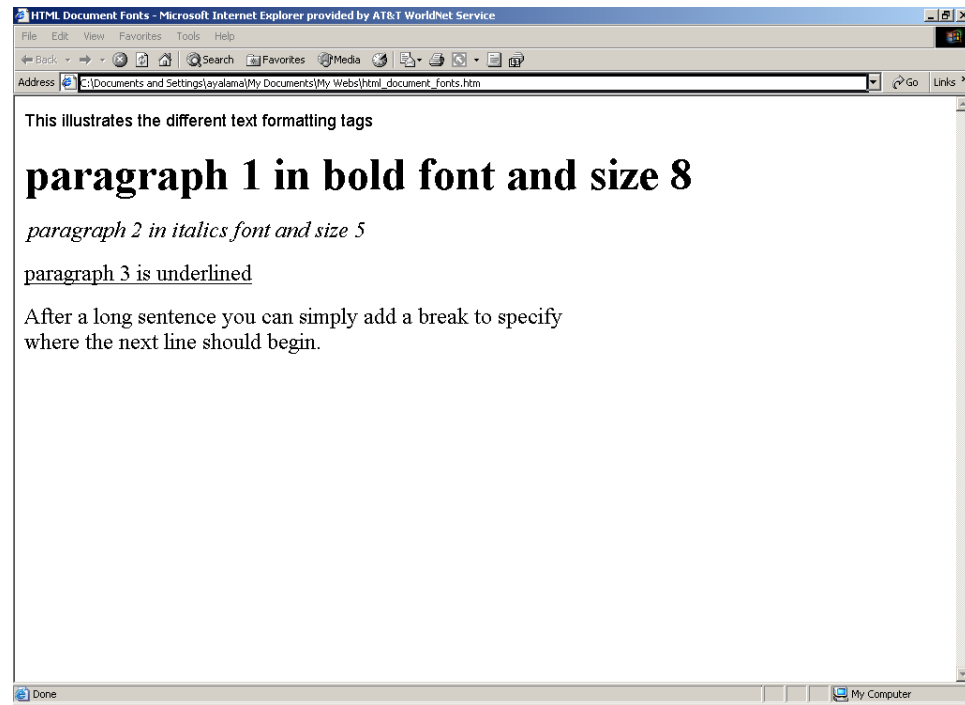


HTML BASICS

MSTP

Text Formatting

- Underline `<u>...</u>`
 - This tag tells the browser to underline the text contained between the `<u>` and the corresponding `</u>` tag. The underlining technique is simplistic, drawing the line under spaces and punctuation as well as the text.
- Font size `<SIZE>...</SIZE>`
 - Changes the tag text font size to the specified size. It is a number.
 - `size=value`





HTML BASICS

MSTP

Tables

- Tables are a great way for format information meaningfully.
- Can be used to structure the layout of a Web page.
- Provides a structured format for presentation of information using the three basic components to define a simple table:
 - Appearance
 - Rows
 - Columns

HTML

MSTP



800 x 600

1024 x 768

- By using tables to format the layout of information flow, you can avoid Browser resolution.

04/10/04

16



HTML BASICS

MSTP

Tables

- Table Tag <Table>...</Table>
 - Defines a table

```
<table width="40%" height="70%" border="2" cellspacing="3" cellpadding="2" >
```

```
<tr align=center valign="center" >  
  <td> Row 1, Col 1 </td>  
  <td> Row 1, Col 2 </td>  
</tr>  
<tr align=center valign="center" >  
  <td> Row 2, Col 1 </td>  
  <td> Row 2, Col 2 </td>  
</tr>  
</table>
```

Row 1, Col 1	Row 1, Col 2
Row 2, Col 1	Row 2, Col 2



HTML BASICS

MSTP

Tables

- Row `<TR>...</TR>`
 - Allows the creation of rows in a table.
 - The following attributes may be defined for a Table Row:

- Row Alignment
- Vertical Alignment
- Background Color

`<table>`

`<tr>`

`</tr>`

`</table>`

← Row definition



HTML BASICS

MSTP

Tables

- Column `<TD>...<TD>`
 - Allows the creation of columns.
 - Like the row, following attributes may be defined for a Table Column:
 - Column Alignment `<HTML>`
`<BODY>`
 - Vertical Alignment `<table width="40%" height="70%" border="2" cellspacing="3" >`
 - Background Color `<tr align=center valign=center >`
 - Heading
 - `<td> Row 1, Col 1 </td>`
 - `<td> Row 1, Col 2 </td>`
 - `</tr>`
 - `<tr align=center valign=center >`
 - `<td> Row 2, Col 1 </td>`
 - `<td> Row 2, Col 2 </td>`
 - `</tr>`
 - `</table>`
 - `</BODY>`
 - `</HTML>`

Column
definition



HTML BASICS

MSTP

Tables

```
<table bgcolor="gray" width="43%" height="70%" border="10"
  cellpadding="15" cellspacing="8" bordercolor="blue" >

  <tr align=center valign=top bgcolor="yellow">
    <td valign="middle" align="center"> <b> Row1, Col 1 <b> </td>
    <td valign="middle" align="center"> <b> Row 1, Col 2 <b> </td>
  </tr>

  <tr align=center valign=top bgcolor="yellow">
    <td valign="middle" align="center" > <b> Row 2, Col 1 <b> </td>
    <td valign="middle" align="center" > <b> Row 2, Col 2 <b> </td>
  </tr>

</table>
```

Row1, Col 1	Row 1, Col 2
Row 2, Col 1	Row 2, Col 2

The above code will produce a table like the one on the left.



HTML BASICS

MSTP

Tables

- Column Heading `<TH>...</TH>`
 - Column headings can be supplied with `<TH>...</TH>` tags appearing within the first row of the table.
 - Centers and bolds the enclosed text over the associated column.

```
<table border="1">  
<caption><b>This is My Table</b></caption>  
<tr bgcolor="silver">  
  <th>Column 1</th>  
  <th>Column 2</th>  
  <th>Column 3</th>  
</tr>
```

This is My Table		
Column 1	Column 2	Column 3



HTML BASICS

MSTP

Tables

`<TABLE>...</TABLE>`

ALIGN = "LEFT | CENTER | RIGHT"
HSPACE = "n"
VSPACE = "n"
BORDER = "n"
BORDERCOLOR = "color name | #rrggbb"
BORDERCOLORDARK = "color name | #rrggbb"
BORDERCOLORLIGHT = "color name | #rrggbb"
BGCOLOR = "color name | #rrggbb"
BACKGROUND = "URL"
CELLPADDING = "n"
CELLSPACING = "n"
WIDTH = "n | n%"

- In general, the Table Tag
`<Table>...</Table>`
syntax:

A number

`<table>`

Number or percent

`<TR>...</TR>`

ALIGN = "LEFT | CENTER | RIGHT"
VALIGN = "TOP | MIDDLE | BOTTOM"
BGCOLOR = "color name | #rrggbb"

`<TD>...</TD>`

`<TH>...</TH>`

ALIGN = "LEFT | CENTER | RIGHT"
VALIGN = "TOP | MIDDLE | BOTTOM"
BGCOLOR = "color name | #rrggbb"
BACKGROUND = "URL"
COLSPAN = "n"
ROWSPAN = "n"
NOWRAP

`<CAPTION>...</CAPTION>`

ALIGN = "TOP | BOTTOM"

04/10/04



LAB 2: TABLES

MSTP

- In this lab you will create a simple table.
 - Type the following code and save it with an .htm extension:

```
<HTML>
<BODY>
  <table border="1" width="251">
    <tr>
      <td width="70"><b>Login:</b></td>
      <td width="70"><input name="login"></td>
    </tr>
    <tr>
      <td width="70"><b>Password:</b></td>
      <td width="70"><input type="password" name="password"></td>
    </tr>
  </table>
</BODY>
</HTML>
```

04/10/04



HTML BASICS

MSTP

Lists

- HTML also contains tags to format bulleted and numbered lists
- Three type of list
 - Unordered
 - Ordered

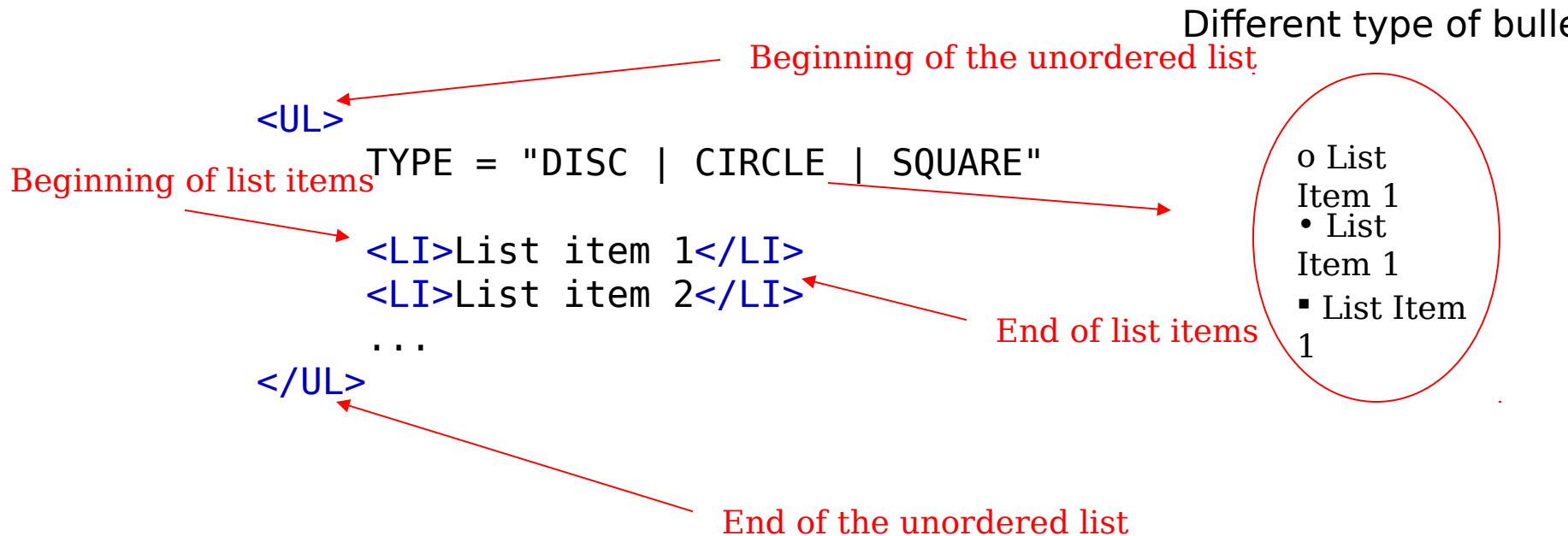


HTML BASICS

MSTP

Lists

- Unordered List
 - The Tag
 - Used to create an unordered list. The list item is identified by the tag.





HTML BASICS

MSTP

Lists

- Ordered List
 - The Tag
 - Used to create an ordered list. The list item is identified by the tag.
 - The TYPE attribute specify one of the five different numbering characters:
 - "1" for Arabic numerals (the default)
 - "A" for upper-case letters
 - "a" for lower-case letters
 - "I" for upper-case Roman numerals
 - "i" for lower-case Roman numerals

Example of ordered list:

```
<ol type="1">
```

```
1.List Item 1  
2.List Item 2
```




HTML BASICS

MSTP

Forms

- Forms provide a way for users to interact with Web pages.
- Data capture device.
- The submitted data may be used:
 - to direct visitors to a different page, much like what happens when clicking a link.
 - to present visitors with personalized pages containing information and links pertinent to their interests or preferences.
 - to trigger a complex search process to locate information or services about which the user is interested.
 - to generate automated email responses.



HTML BASICS

MSTP

Forms

- Forms gather information from users by displaying special form fields that permit the user to enter data or make selections.
- Standard controls that can be coded on a Web form are:

Text Box:

Password:

Textarea:

Radio Button: ☐ Radio Button

Checkbox: ☒ Checkbox

Selection Menu:

Submit Button:

Reset Button:



HTML BASICS

MSTP

Forms

- The <FORM> Tag
 - General format

```
<form name="form name" action="URL" method="GET | POST">
```

```
</form>
```

- It has three attributes
 - Name
 - Action
 - Method



HTML BASICS

MSTP

Forms

- Name attribute
 - Assign the name of the form so it can be referenced in a script.
- Action attribute
 - Identifies the page to which the form is submitted.
- Method attribute
 - Specifies the manner in which form information will be submitted. Two possible values:
 - GET: information is submitted appended to the Action URL.
 - POST: information is transmitted as a separate data stream.



HTML BASICS

MSTP

Forms

```
<html>
<head>
    <title>A Form Page</title>
</head>
<body>
    <form name="MyForm" action="AspPage.asp" method="POST">
        ...
    </form>
</body>
</html>
```

The name of this form is MyForm. The information gathered by this form will be submitted to AspPage.asp using the method POST.



HTML BASICS

MSTP

Forms

- `<INPUT TYPE="TEXT">` Tag
 - The most commonly encountered type of form field.
 - Presents a standard text entry box into which information can be typed.
 - Format:
 - `<INPUT>`
 - `TYPE = "TEXT | PASSWORD"`
 - `NAME = "field name"`
 - `SIZE = "n"`. Determines the textbox in characters. Default is 20 characters.
 - `MAXLENGTH = "n"`. Determines the maximum number of characters that the field will accept.
 - `VALUE = "text string"`. Will display its contents as the default value.

`<form>`

Last Name: `<input type="text" name="LastName">`

`</form>`

Last Name

04/10/04



HTML BASICS

MSTP

Forms

- **<INPUT TYPE = "PASSWORD"> Tag**
 - Similar in function to a text field
 - All the attributes associated with the text field are applicable to the password type.

<form>

Password: <input type="password" name="Password">

</form>

Password:
d:

The name of the field is Password.
It is up to you to assign the name.



HTML BASICS

MSTP

Forms

- **Drop-Down Lists**

- Presents items that are chosen from a drop-down list.
- One or more items can be chosen by clicking the entry.

```
<form>
```

```
...
```

```
How do you like to travel?<br>
```

```
<select name="Mode">
```

```
<option value="1">Airplane</option>
```

```
<option value="2">Car</option>
```

```
<option value="3">Bus</option>
```

```
<option value="4">Ship</option>
```

```
</select>
```

```
...
```

```
</form>
```

How do you like to travel?



HTML BASICS

MSTP

Forms

- The `<INPUT TYPE = "SUBMIT">` Tag
 - All forms must include at least one "submit" button to submit the form information for processing.

```
<form>
```

```
...
```

```
<input type="submit" name="SubmitButton" value="Submit Query">
```

```
...
```

```
</form>
```




HTML BASICS

MSTP

Forms

- The `<INPUT TYPE = "RESET">` Tag
 - Permit users to clear all information from a form.



```
<form>
...
<input type="reset" value="Reset">
...
</form>
```


INTERNET INFORMATION SERVER



MSTP

INTERNET INFORMATION SERVER OVERVIEW

04/10/04

37

GETTING STARTED



MSTP

IIS Overview

- ASP files do not run in your browser like HTML.
- ASP code is evaluated on the server before you see it in your Web browser.
- You need Internet Information Server (IIS) installed.
 - Version 5.0 comes with Windows 2000.

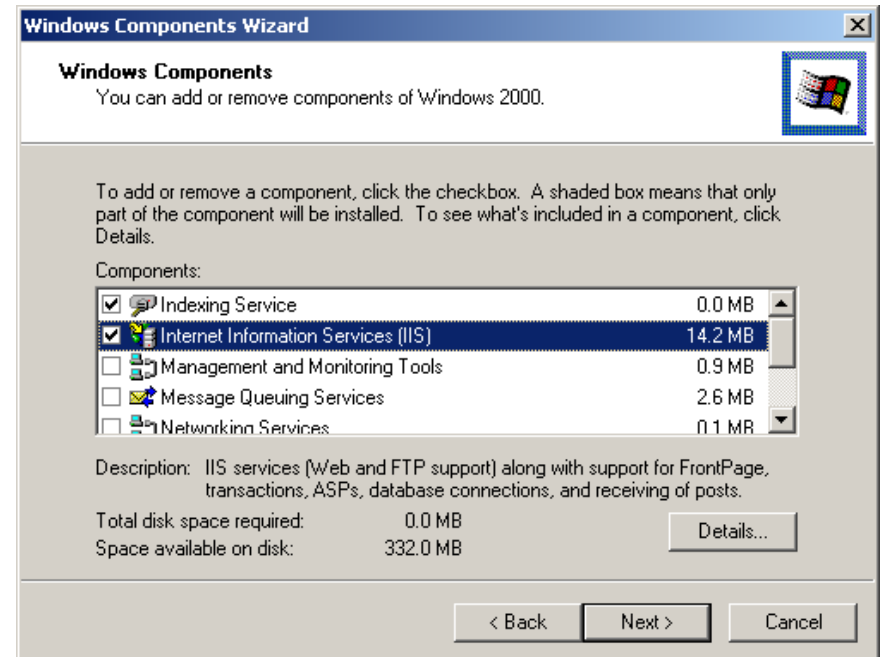
GETTING STARTED



MSTP

IIS Overview

- Installing IIS
 - If you are installing IIS after completing the Windows installation, go to Control Panel ⇒ Add/Remove Programs, and pick the Add/Remove Windows Components button.
 - This is performed by your systems administrator.



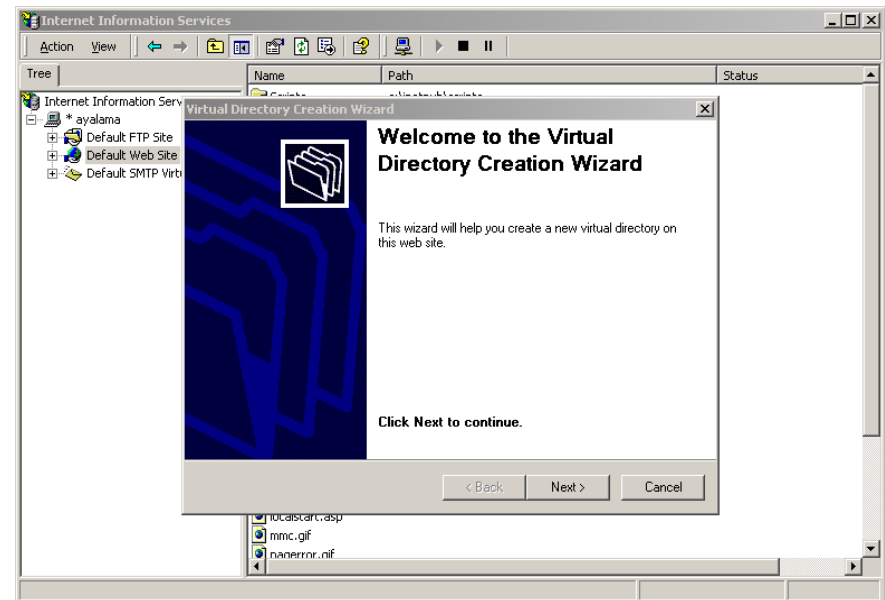


GETTING STARTED

MSTP

Virtual directories

- Creating virtual directories
 - Two ways of creating virtual directories:
 - Using the wizard.
 - Not using the Wizard.
 - Using the Wizard
 - From the IIS Manager, right-click the Default Web Site and select New ⇒ Virtual Directory from the pop-up menu.



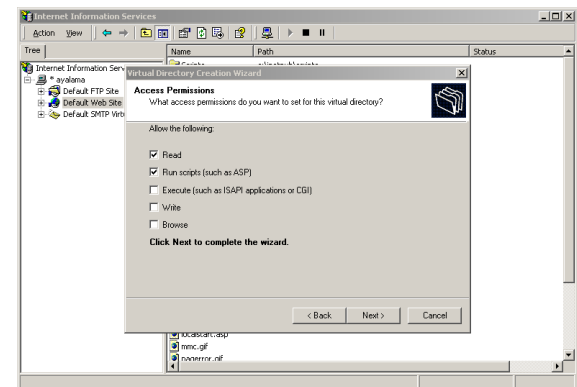
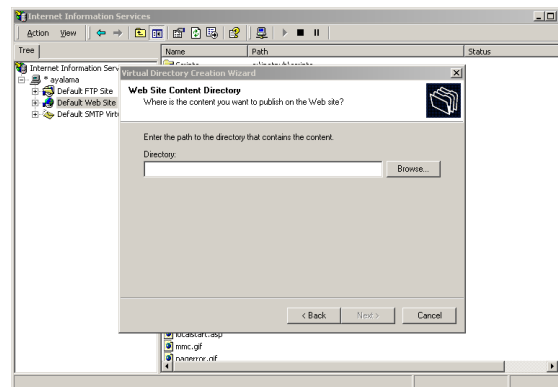
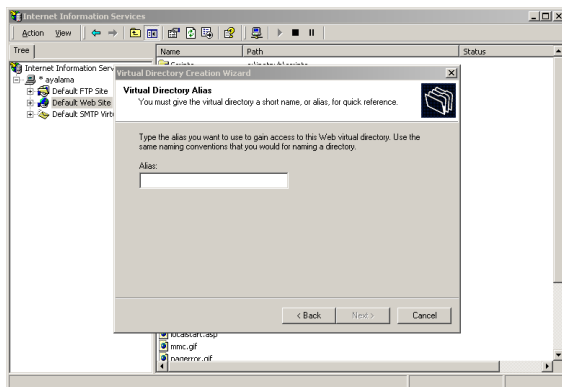
GETTING STARTED



MSTP

Virtual directories

- Follow the next three steps:
 - You need to specify the name or alias you want to use in your Web browser.
 - Specify the physical directory the alias or virtual directory will be pointed at.
 - Set the access permissions.



04/10/04

ASP



MSTP

ASP OVERVIEW

04/10/04

42

ASP OVERVIEW



MSTP

- ASP is designed to let you create pages that can change each time a user loads them.
- You write code in the page that is run on the Web server before the user sees the page.
- ASP enables server side scripting for IIS with native support for both VBScript and JScript.



ASP OVERVIEW

MSTP

Composition

- Active Server Pages are text based files comprised of a combination of HTML tags and Active Server scripts.
- ASP Delimiters
 - Active Server scripts are distinguished on the page from HTML tags by using `<%` and `%>` delimiters. The delimiter can be embedded within HTML tags.
- Setting the ASP Scripting Language
 - To set the scripting language:
`<%@ LANGUAGE = "VBScript" %>`
- Variables, Operators, and Statements
 - Each scripting language has its own specific syntax that is used to define and set variables, use operators for comparing items, and use statements to help define and organize the code.



ASP OVERVIEW

MSTP

Composition

- Active Server Components and Objects
 - The scripting variables, operators, and statements can be used to tap into special Active Server tools that add programming functionality to the server. These tools consist of Active Server Objects components. The most commonly used objects are:
 - **Request:** responsible for retrieving information from the browser.
 - **Response:** responsible for sending information to the browser
 - **Session:** responsible for managing information for a specific user.
 - **Server:** responsible for administrative functionality of the browser.



ASP OVERVIEW

MSTP

Composition

```
<SCRIPT Language="VBScript" RUNAT="Server">  
Subroutine  
...  
</SCRIPT>  
  
<HTML>  
<HEAD>  
  <TITLE>Web Page</TITLE>  
</HEAD>  
<SCRIPT Language="JavaScript">  
...  
</SCRIPT>  
<BODY>  
  <%  
In-line processing commands  
...  
SQL command  
%>  
<TABLE>  
<TR>  
  <TD><%=field%></TD>  
  <TD><%=field%></TD>  
</TR>  
</TABLE>  
<!-- INCLUDE File="file.inc" -->  
</BODY>  
</HTML>  
  
<SCRIPT Language="JavaScript">  
function {  
...  
}  
</SCRIPT>
```

VBScript which runs on the server as a subroutine called from another script.

HTML that is interpreted by the client when the page is returned from the server.

JavaScript in-line routine that runs on the client as the page is loaded.

VBscript which runs on the server with embedded SQL call to database server.

HTML for client display of embedded data values from the server

Server-side Include to copy external HTML file into page before returning to client.

JavaScript function that is called by the client after page is formatted by the browser.



ASP OVERVIEW

MSTP

Running ASP Scripts

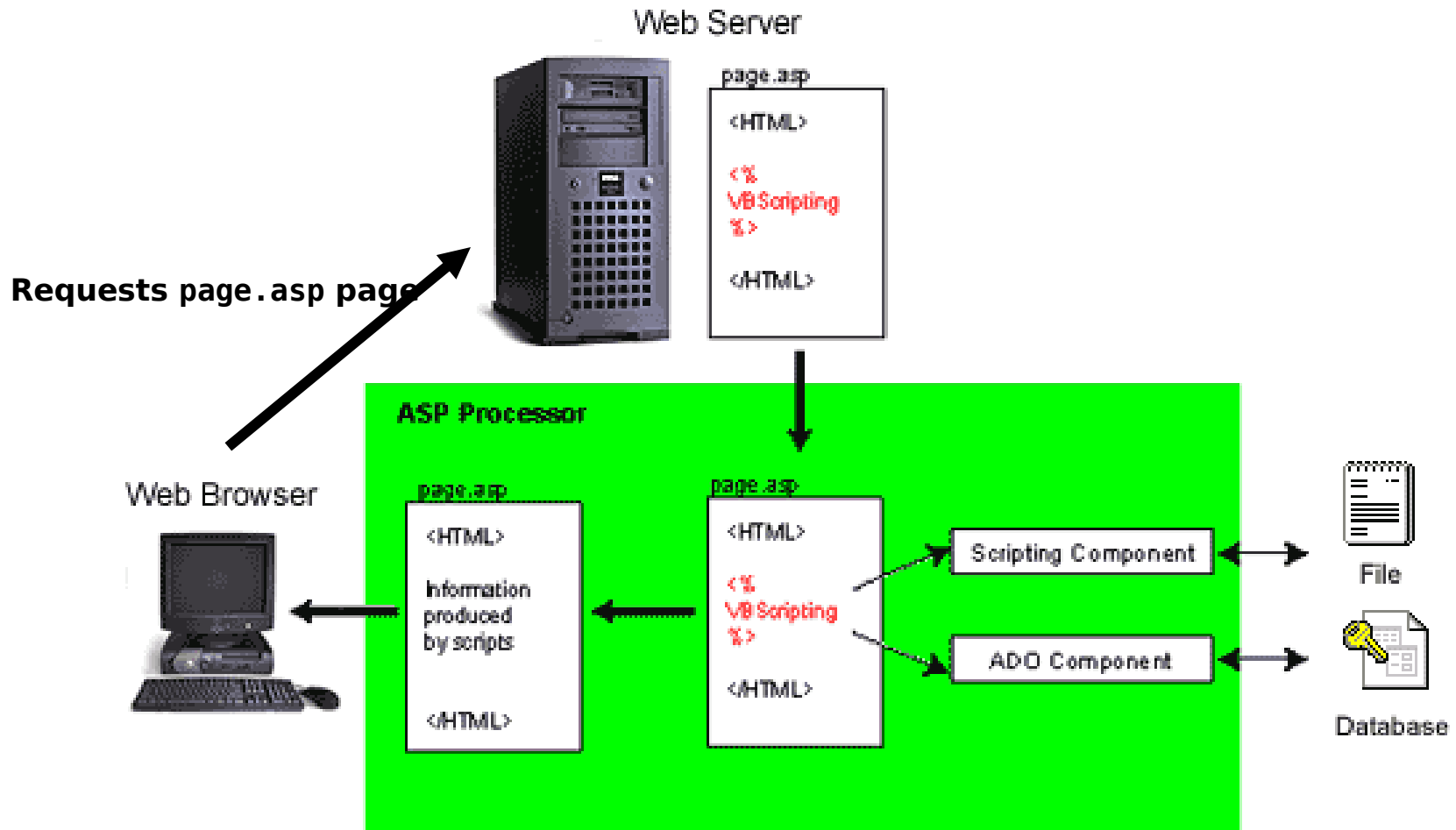
- Pages must be saved with the file extension .asp to inform the server that this is a scripted page.
- The page is processed by the ASP processor where the script is run and the information produced by the script is generated.
- The server generates an HTML page with the information generated by the script (to be sent to the browser).
- End result is a page composed entirely of HTML and text information generated by the script.



ASP OVERVIEW

MSTP

Running ASP Scripts



VBSCRIPT



MSTP

VBSCRIPT INTRODUCTION

04/10/04

VBSCRIPT



MSTP

- Differences between Visual Basic and VBScript
 - Subset of VB.
 - VBScript does not have a design-time environment like VB.
 - VBScript code "lives" within an HTML document, which is a plain text file. VBScript code works inside of HTML documents and runs along with HTML.
 - Visual Basic code creates Windows applications that operate in and of themselves.
 - Visual Basic supports many commands, keywords, and data types that VBScript does not support.



VBSCRIPT

MSTP

Variables and Data types

- Data types
 - Only one, *Variant*.
 - *The duality of a variant*: A Variant can contain either numeric or string (text) information.
 - It behaves like a number when used as a number or as a string when used in a string context.
 - A Variant can contain the following subtypes:
 - Long
 - Single
 - Double
 - Date (Time)
 - String
 - Object
 - Error
 - Empty
 - Null
 - Boolean
 - Byte
 - Integer
 - Currency



VBSCRIPT

MSTP

Variables and Data types

- Variables

- It is a place holder for information.
- Variables can be created in two ways: explicit method, implicit method.
- Explicit Method
 - The explicit method is where you use the Dim keyword to tell VBScript you are about to create a variable. You follow this keyword with the name of the variable.

```
Dim Quantity
```

```
Dim X, Y, Z
```

- Implicit Method
 - You do not use the Dim statement.
 - You can just start using the variable in your code, and VBScript creates it automatically.

```
Quantity = 10
```




VBSCRIPT

MSTP

Variables and Data types

- Explicit is highly recommended.
- To make the explicit method a requirement and prevent implicit allocations of names, you must place the command:

Option Explicit

in the first line of the script.

```
<%  
Option Explicit  
Dim Quantity  
...  
%>
```




VBSCRIPT

MSTP

Variables and Data types

- Look at the following code:

```
<%  
...  
Quantity = 2  
Quantity = Quantite + 3  
...  
%>
```

- What would be the value of Quantity?

Quantity = 3 **Why????????!!!!!!!!!!!!!!!!!!!!**

You could spend hours trying to
find out why!!



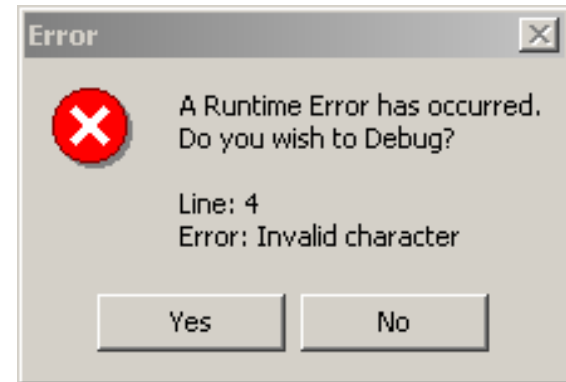
VBSCRIPT

MSTP

Variables and Data types

- Same code with the enforced explicit method

```
<%  
Option Explicit  
Dim Quantity  
Quantity = 2  
Quantity = Quantite + 3  
...  
>%
```



- Quantity is "misspelled"



VBSCRIPT

MSTP

Constants

- Takes place of a number or string that never changes.
- VBScript has several built-in constants.
- You create user-defined constants using the Const statement.

```
Const MY_STRING = "USMC"
```

```
Const USMC_BDAY = "10 Nov 1775"
```

- You may want to adopt a naming scheme to differentiate constants from variables.
 - All your constants in capital letters.

Constant

```
Const MY_STRING = "USMC"
```

Variable

```
My_String = "USMC"
```

04/10/04



VBSCRIPT

MSTP

Control Structure

- Control Structures allows you to control the script based on some conditions.
- The most commonly used control structures in VBScript are:
 - *If...Then...Elseif*: Executes a block of code based on a condition.
 - *Select...Case*: Executes a block of code based on the value of a variable been compared.



VBSCRIPT

MSTP

Control Structures

- If...Then...Else

- Conditionally executes a group of statements, depending on the value of an expression.

- Syntax:

If 'condition' **Then** 'single line of code to be executed'

- Example:

```
<%  
Dim MyDate  
MyDate = #2/13/95#  
If MyDate < Date() Then MyDate = Date()  
%>
```

Date() is a built in function in VBScript
It returns the current date of the system.

MyDate is assigned 13 Feb 1995. Today's date is 21 May 2003.
Since MyDate is less than today's date, MyDate will be assigned 21 May 2003.



VBSCRIPT

MSTP

Control Structures

- You can have a block of statements by using:

```
If 'condition' Then  
    'block of code to be executed'  
ElseIf 'condition-n' Then  
    'block of code to be executed'  
Else  
    'block of code to be executed'  
End If
```

- Adding Elseif clauses expands the functionality, controlling the flow of the program based on several possibilities.

04/10/04

```
<%  
Dim Status, MarineStatus  
Status = "T"  
    If Status = "T" Then  
        MarineStatus = "TAD"  
    ElseIf Status = "L" Then  
        MarineStatus = "On Leave"  
    ElseIf Status = "A" then  
        MarineStatus = "Accounted"  
  
    Else  
        MarineStatus = "UA"  
    End If
```




VBSCRIPT

MSTP

Control Structures

- Select Case

- Provides an alternative to *If...Then...Elseif* for selectively executing one block of statement.
- Single test expression evaluated once at the top of the structure. The result of the expression is compared with the values for each Case in the structure.
- If there is a match, the block associated with that Case is executed.

```
<%  
Dim Status, MarineStatus  
Status = "A"  
Select Case Status  
    Case "A"  
        MarineStatus = "Accounted"  
    Case "L"  
        MarineStatus = "On Leave"  
    Case "T"  
        MarineStatus = "TAD"  
    Case Else  
        MarineStatus = "UA"  
End Select  
%>
```




VBSCRIPT

MSTP

Looping Structure

- Looping allows you to run a group of statements repeatedly.
- The most commonly used looping structures in VBScript are:
 - *Do...Loop*: Loops while or until a condition is True.
 - *While...Wend*: Loops while a condition is True.
 - *For...Next*: Uses a counter to run statements a specified number of times.



VBSCRIPT

MSTP

Looping Structure

- Do...Loop
 - To run a block of statements an indefinite number of times.
 - The statements are repeated either while a condition is *True* or until a condition becomes *True*.
 - You can place a conditional test at either the start or end of the loop structure. This provides you with the ability to force a loop to execute at least one.

```
Do While 'condition'  
    ... code within the loop goes here  
Loop
```

or

```
Do  
    ... code within the loop goes here  
Loop While 'condition'
```




VBSCRIPT

MSTP

Looping Structure

- Do...Loop

```
<%  
Do While result < 50  
    result = result + 1  
Loop  
Response.Write "The value of result Do While = " &result  
%>
```

The condition is tested
before executing the
code.

```
<%  
Do  
    result = result + 1  
Loop While result < 50  
Response.Write "The value of result Loop While= " &result  
%>
```

The condition is tested
after execution of the
code.



VBSCRIPT

MSTP

Looping Structure

- While...Wend
 - Identical to the *Do...While* loop structure.
 - *Wend* takes place of the *Loop* keyword.
 - The *While...Wend* executes until the conditional statement following the *While* becomes True.

```
While 'condition'  
    'do something  
Wend
```

Example:

```
<%  
...  
While result < 50  
    result = result + 1  
Wend  
Response.Write "The value of result = " &result
```

04/10/04



VBSCRIPT

MSTP

Looping Structure

- For...Next
 - Loop a pre-determined number of times.
 - Uses a loop variable to control the number of loops to be executed.

For variable = start **to** end **Step** increment
'code that gets repeated
Next

- The default increment step is one, negative steps can be done.

```
Dim x, result
x = 0
For x = 1 to 50
    result = result + 1
Next
Response.Write "The value of result = "&result
```

The value of result = 50

```
Dim x, result
x = 0
For x = 1 to 50 Step 2
    result = result + 1
Next
Response.Write "The value of result = "&result
```

The value of result
= 25

04/10/04

ASP 3.0



MSTP

ACTIVE SERVER PAGES (ASP) 3.0

ASP



MSTP

Processing user input

- Request Object
 - You can think of it as the *input* object.
 - Holds the information sent from the browser to the web server.
 - Responsible to make that information available to the ASP application.
 - How to get data from the user to the server?
 - HTML forms.

ASP

MSTP

Processing user input

- A form is the primary way to obtain information from the user.

Form's name

Page that will process the data received.

```
<form method="POST" name="LoginForm" action="check_password.asp">
<table border="2" width="400">
  <tr>
    <td width="91"><b>Login:&nbsp;</b></td>
    <td width="295"><input name="login"></td>
  </tr>
  <tr>
    <td width="91"><b>Password:</b></td>
    <td width="295"><input type="password" name="password"></td>
  </tr>
</table>
<p><input type="submit" value="Login" name="submit"> <input type="reset"
value="Reset" name="Reset"></p>
</form>
```

Method *POST* so Request object will be capable of accessing the information.

Name of the text box. Request object can ask for the information.

04/10/04



ASP

MSTP

Processing user input

Form structure showing input fields and buttons:

Login: `<input name="login">`

Password: `<input type="password" name="password">`

Login Reset

- Two text boxes to get information.
- To get the information from the form use the Request object.

We are getting the information from the form and assigning it to two local variables.

```
<%  
...  
password = Request.Form("password")  
login = Request.Form("login")  
...  
%>
```

```
<%  
...  
password = Request ("password")  
login = Request ("login")  
...  
%>
```


ASP



MSTP

Session Object

- Represents the current user's session on the web server.
- It is user specific.
- Its properties and methods allow you to manipulate the information on the server that is specific to that user for the duration of that user's connection.
- The most common use is for session variables.

ASP



MSTP

Session Object

- Session variables
 - Declared similar to application variables
 - To declare or reference one, use the Session object and name of the variable.
 - `Session (variableName)`
 - To add a Session variable, simple assign a value to a session variable name.
`Session ("Permissions") = "user"`
 - To clear a Session variable, set it to a null string ("") or set it to the VBScript *Empty* value
`Session ("Permissions") = ""`
`Session ("Permissions") = Empty`

ASP



MSTP

Responding to the User

- Response Object
 - If the *Request* object is the *input* object, then the *Response* object is just the opposite, the *output* object. The *Response* object allows you to send information from the Web server to the browser.
 - Probably the most important methods of the *Response* object are the *Redirect* method, and the *Write* method.

ASP



MSTP

Responding to the User

- Response.Write

- The *Write* method is used to send output to the browser.
- Syntax:

Response.Write *variant*

- Can contain literal text strings (enclosed in quotes), values retrieved from VBScript functions along with HTML tags (enclosed in quotes as part of a text string) to control output formatting.

Response.Write("text string" | & | server value | "<HTML tag>")

- For example, to display the time a page was requested by using the *Write* method, we could use the following syntax:

<% *Response.Write* Now %> or <% =Now %>

Same thing



ASP

MSTP

Responding to the User

- To have HTML tags and text:

```
<%  
...  
Response.Write("<b>The current date is " & Date())  
Response.Write(" and the current time is " &  
    Time()&"</b>")  
...  
>%
```

Bold tag

The current date is 4/10/2004 and the current time is 11:18:19 PM.

ASP



MSTP

Responding to the User

- Alternatively, the script can be written as a single line of code:

```
<%
```

```
...
```

```
Response.Write("<b>The current date is " & Date() &  
" and the current time is " & Time() & ".</b>")
```

```
...
```

```
%>
```

The current date is 4/10/2004 and the current time is 11:18:19 PM.

ASP



MSTP

Responding to the User

- Response.Redirect
 - Tells the browser where to go.

`Response.Redirect (http://rightpage.htm)`

- Useful when making decisions based on a user's actions.

```
<%  
"  
If Session("Permissions") = "user" Then  
    response.redirect "Show_Records.asp"  
ElseIf Session("Permissions") = "admin" Then  
    response.redirect "Show_RecordsAdmin.asp"  
Else  
    response.redirect "login.asp"  
End If
```

```
"  
%> 04/10/04
```


DATABASE OVERVIEW



MSTP

INTRODUCTION TO DATABASE

04/10/04

77



DATABASE OVERVIEW

MSTP

Planning a Database

- The purpose of planning a database is to identify:
 - The information you currently track.
 - The information you want or need to track in the future.
 - The reports you need to produce.
- Database Structure
 - You need to design a structure for the database.
 - You need to make sure your database is a Relational Database.
 - Each database file has the following elements:
 - Field: one portion of the data, also known as column.
 - Record: related information, also known as row. A single record is made up by one or more fields.
 - Database Table: made up by one or more records.



DATABASE OVERVIEW

MSTP

Planning a Database

- Relational Database Tables

Names	
ID	
SSN	
FName	
LName	
MI	
Rank	
Unit	
Login	
Password	
Permissions	
Phone	
Spouse	
SpouseMI	

Kids	
KidIdNum	
SponsorSSN	
FName	
Age	

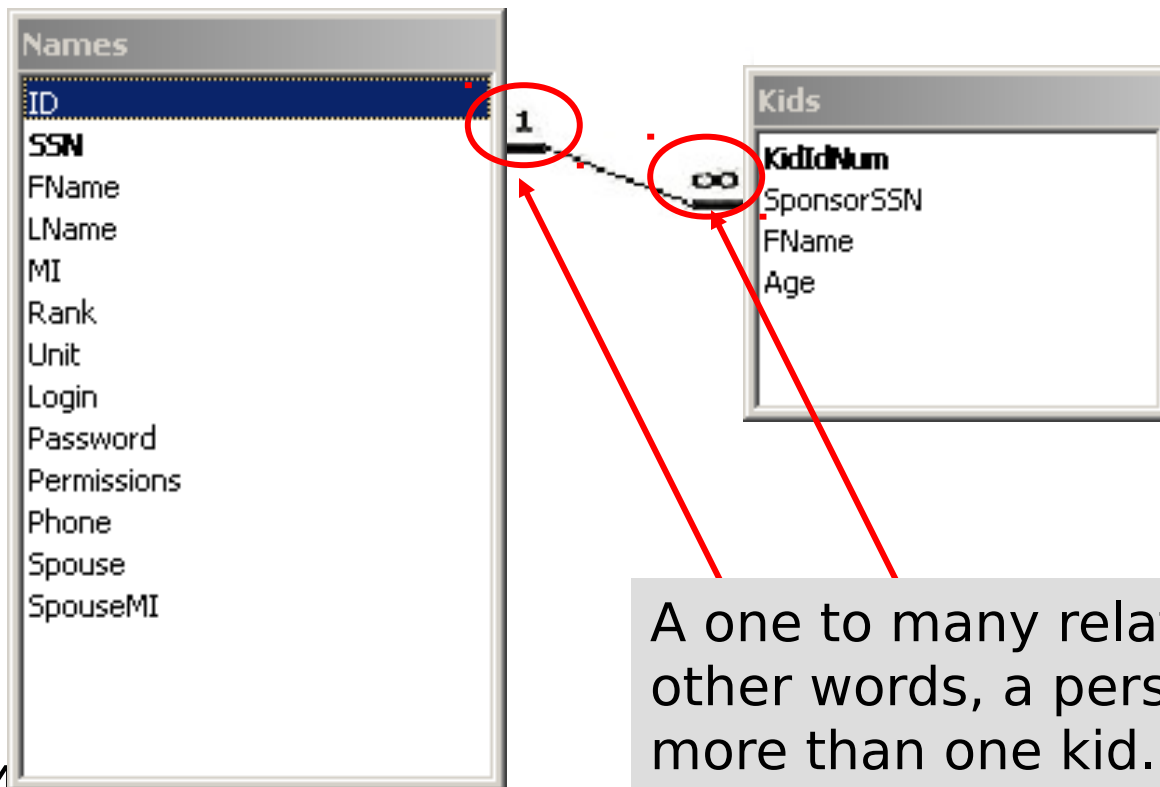


DATABASE OVERVIEW

MSTP

Planning a Database

- Relational Database
 - Tables Relationships



A one to many relationships, in other words, a person can have more than one kid.

DATABASE OVERVIEW



MSTP

Fields

Planning a Database



Table structure

Field Name	Field Type	Field Size
ID	AutoNumber	Long Integer
SSN	Number	Long Integer
FName	Text	15
LName	Text	15
MI	Text	2
Rank	Text	10
Unit	Text	50
Login	Text	15
Password	Text	15
Permissions	Text	10
Phone	Text	15
Spouse	Text	15

- Fields are the data holders.
- The record shown has 12 fields.

Fields

One Record



ID	SSN	FName	LName	MI	Rank	Unit	Login	Password	Permissions	Phone	Spouse
1	123456789	Miguel	Ayala	A	Capt	MSTP	ayalama	ayalama	admin	7037846001	Daisy

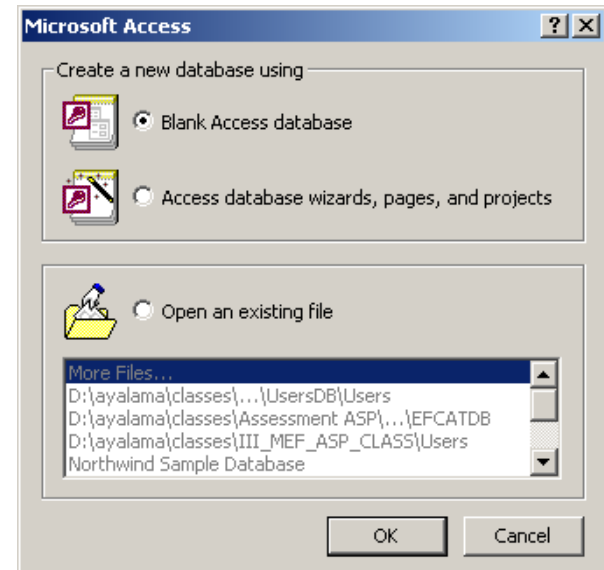
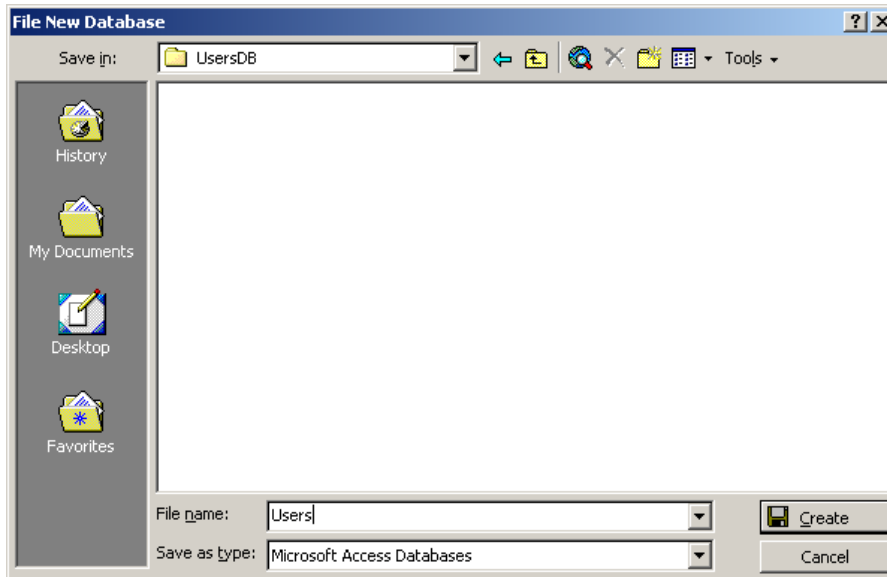


DATABASE OVERVIEW

MSTP

Building a Database with Microsoft Access

- Creating a Database
 - When you open Microsoft Access you will see the dialog box at the right. Click the "Blank Database" button to indicate that you are creating a new



- When the "File New Database" dialog box appears, name it with your Users.mdb and save it in your directory on the server by clicking the "Create" button.

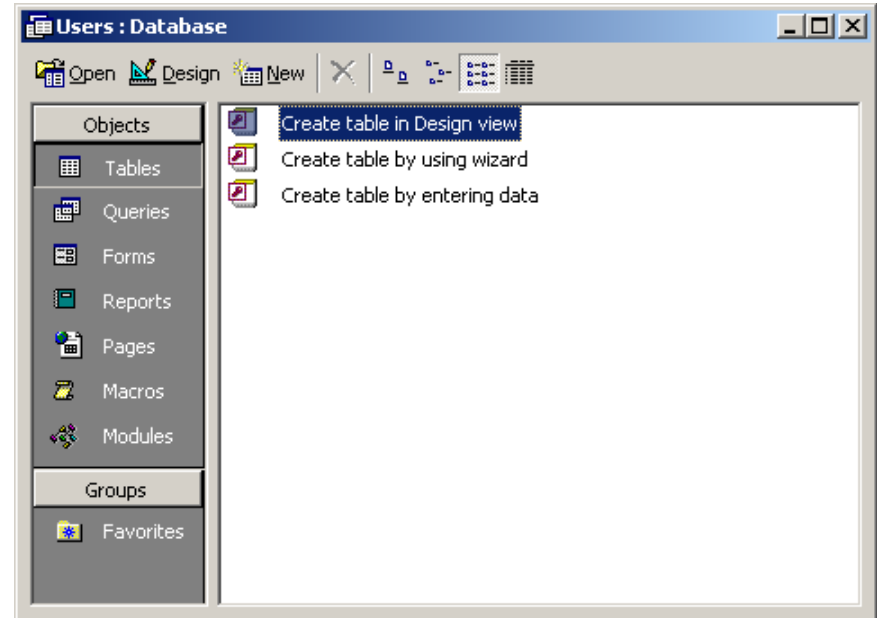


DATABASE OVERVIEW

MSTP

Building a Database with Microsoft Access

- Creating a Table
 - You are presented with a set of options for creating or selecting tables and other components of your database. You can create your table using the Wizard, in Design View or by entering data into generic fields. We will do it in Design View. Click the *Design* button.





DATABASE OVERVIEW

MSTP

Building a Database with Microsoft Access

- Field Specifications
 - Assign a Field Name (cannot include blank spaces).
 - Select the Data Type of the field (Text, Memo, etc.).
 - Specify the Field Size characters.
- Assign it a name and save the database.

The screenshot shows the 'Names : Table' window in Microsoft Access. It contains a table with the following fields:

Field Name	Data Type	Description
ID	AutoNumber	
SSN	Number	
FName	Text	
LName	Text	
MI	Text	
Rank	Text	
Unit	Text	
Login	Text	
Password	Text	
Permissions	Text	
Phone	Text	
Spouse	Text	

Below the table is the 'Field Properties' section. The 'General' tab is selected, showing the following properties:

Property	Value
Field Size	Long Integer
New Values	Increment
Format	
Caption	
Indexed	Yes (No Duplicates)

A note on the right side of the 'Field Properties' section states: 'A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.'



DATABASE OVERVIEW

MSTP

Building a Database with Microsoft Access

- Adding Information into the Database
 - Open the table by double-clicking it.
 - The datasheet view presents the columns and rows of the table where you can enter your data.
 - Type the information in the columns and tab to the next one.
 - When you are finished, you would have created one new record.

The screenshot shows the Microsoft Access interface in Datasheet View. The title bar reads 'Microsoft Access - [Names : Table]'. The menu bar includes File, Edit, View, Insert, Format, Records, Tools, Window, and Help. The toolbar contains various icons for file operations, editing, and navigation. The datasheet table has the following columns: ID, SSN, FName, LName, MI, Rank, Unit, Login, Password, Permissions, Phone, and Spouse. The first record shows ID 55, SSN 123-45-6789, FName Miguel, LName Ayala, MI A, Rank Capt, Unit MSTP, Login ayalama, Password ***** (masked), Permissions admin, Phone (540) 784-6001, and Spouse Daisy. A second record is partially visible with ID 56, SSN 123-45-6789, and FName Miguel. The status bar at the bottom indicates 'Record: 1 of 1' and 'Datasheet View'.

ID	SSN	FName	LName	MI	Rank	Unit	Login	Password	Permissions	Phone	Spouse
55	123-45-6789	Miguel	Ayala	A	Capt	MSTP	ayalama	*****	admin	(540) 784-6001	Daisy
56	123-45-6789	Miguel									

INTRODUCTION TO STRUCTURED QUERY LANGUAGE (SQL)



MSTP

STRUCTURED QUERY LANGUAGE (SQL)



INTRO TO SQL

MSTP

Structured Query Language (SQL)

- Easy and faster way to extract information from a database table.
- You are relying on the Database Management System (DBMS) to perform the work rather than coding a server script to access tables.
- The script simply issues a request to the DBMS, which independently carries out the task.
- Faster and more reliable.



INTRO TO SQL

MSTP

SQL Statements

- The SELECT Statement
 - This statement is used to select records from a database table.
 - The selection can encompass the entire table with all of its fields, or it can be restricted to certain fields in certain records matching a given criteria.
 - The group of selected records itself becomes a recordset that can be processed in the same fashion as used for an entire table.

```
SELECT  * | field1[,field2]...  
FROM TableName  
WHERE condition  
ORDER BY field1 [ASC/DESC] [, field2 [ASC/DESC] ]
```




INTRO TO SQL

MSTP

SQL Statements

- The keyword *SELECT* is followed by one of two specifications identifying the fields of data to be selected from a table.
 - An asterisk (*) denotes that ***all*** fields are to be selected for each record.
 - You can provide a list of field names, separated by commas, and only those data fields will be selected.
- The *FROM* clause identifies the table from which these records and fields are to be selected.

SELECT * FROM MyTable

Selects all records from MyTable and includes all (*) of the fields that make up a record.

SELECT LastName,FirstName FROM MyTable

Selects all records and fields from MyTable, but only provides the specified fields.



INTRO TO SQL

MSTP

SQL Statements

- The WHERE Clause
 - Used when you do not want or need to retrieve all the records in a table but just those who match certain condition.
 - The keyword *WHERE* is followed by one or more selection criteria. A common way of using this feature is to check for equality, that is, to look for a matching value in one of the record's fields.

```
SELECT * FROM Marines WHERE Base='Quantico'
```
 - The DBMS would deliver those records that had a matching criteria (Marines from Quantico).



INTRO TO SQL

MSTP

SQL Statements

- Common conditional operators:
 - = (equal to)
 - <> (not equal to)
 - < (less than)
 - > (greater than)
 - <= (less than or equal to)
 - => (equal to or greater than)
- You can combine tests using logical operators like AND, OR, and NOT to expand or contract your selection.

```
SELECT * FROM Marines WHERE base='Quantico' OR base='Camp  
Lejeune'
```

04/10/04



INTRO TO SQL

MSTP

SQL Statements

- The ORDER BY Clause
 - Use to arrange, or sort, the set of records retrieved from the table.
 - Identifies the names of fields on which to sort the records.
 - If more than one field name is supplied, then sorting takes place in the order in which the names appear, separated by commas. The first field becomes the *major* sort field, the second field becomes the *intermediate* sort field, and the third field becomes the *minor* sort field.

```
SELECT * FROM Marines  
ORDER By LastName,FirstName,MiddleInitial
```


ASP



MSTP

MORE ASP

04/10/04

ASP



MSTP

Accessing a Database

- The Connection Object
 - Before you can retrieve any data from a database, you have to create a connection to that database.
 - The *Connection Object* contains the properties and methods necessary to make a link between a Web page and a database.
 - The method for creating a Connection Object is to use the VBScript **Set** statement, calling upon the Server Object to create a Connection Object for our script.

```
Set ConnectionObject=Server.CreateObject("ADODB.Connection")
```




Accessing a Database

- ActiveX Data Objects (ADO) Constants
 - The ADO are a set of objects that you can use to access databases. Each constant represents a numeric value
 - For example, in the *adovbs.inc* you will find the four constants for the recordset type defined as follows:

```
Const adOpenForwardOnly = 0
Const adOpenKeySet = 1
Const adOpenDynamic = 2
Const adOpenStatic = 3
```
 - You can use the *#include* directive to read another page in and make it part of the current page.
 - In order to have those constants available to your page, you have to include the file *adovbs.inc*.

```
<!--#include file="adovbs.inc"-->
```


ASP



MSTP

Accessing a Database

- Creating an Open Database Connectivity (ODBC) connection.
 - Allows programs to access different kinds of databases in almost the exact way.
 - The method for creating a Connection Object is to use the VBScript *Set* statement and calling upon the Server Object to create a Connection Object for our script.

<%

...

Dim ConnectionObjectName

Set ConnectionObjectName =

Server.CreateObject("ADODB.Connection")

...

%>

Name of the connection object
(you name it)

ASP



MSTP

Accessing a Database

- ODBC Data Source Name (DSN)-less Connection

<%

...

Dim dsn

Dim Conn

Set Conn = Server.CreateObject("ADODB.Connection")

dsn="DBQ=" & Server.MapPath("UsersDb/Users.mdb") &
";Driver={Microsoft Access Driver (*.mdb)}";

...

%>

To avoid writing absolute path use the Server.MapPath method. This will accept a relative or virtual path and returns a physical path. Good when you do not have control of the server.

The database file is located under that folder in the same directory as the application.



Accessing a Database

- Connecting to a Database
 - Once you have a *Connection* object already created, you are ready to connect to the database.

<%

...

ConnectionObjectName.Open dsn

...

%>




Accessing a Database

- The Recordset Object
 - The *ASP Recordset Object* contains the properties and methods necessary to extract data from a database table and to make that set of records available to a script.
 - The general format for creating a Recordset Object is similar to the method used to create a Connection Object:

```
<%  
...  
Dim RecordsetObjectName  
Set RecordsetObjectName =  
Server.CreateObject("ADODB.Recordset")  
...  
>%
```

Name of the recordset object
(you name it)

A red arrow points from the text "Name of the recordset object (you name it)" to the variable name "RecordsetObjectName" in the code.



ASP

MSTP

Accessing a Database

- The Beginning of File (BOF) Object
 - If the value of the BOF property of a Recordset object is True, the current record pointer is positioned one record before the first record in the recordset.
 - This is a read-only property.
 - You can use the BOF property in conjunction with the End of File (EOF) property to ensure that your recordset contains records and that you have not navigated beyond the boundaries of the recordset.

<%

...

If Not rs.BOF Then

' There are records. Use the EOF property to loop
' through all the records in the recordset and
' display them to the screen.

...

%>

We named our Recordset Object rs



Accessing a Database

- The End of File (EOF) Object

- If the value of a Recordset object's EOF property is True, the current record pointer is positioned one record after the last record in the recordset.
- This is a read-only property.
- Note that the value of EOF is also True if there are no records in the recordset.

```
<%  
    Do While Not rs.EOF  
%>  
  
<tr>  
  
    <td width="63"><%=rs("Rank")%> </td>  
    <td width="158"><%=rs("FName")%> </td>  
    <td width="41"><%=rs("MI")%> </td>  
    <td width="113"><%=rs("LName")%> </td>  
    <td width="54"><%=rs("Unit")%> </td>  
    <td width="152"><%=rs("Phone")%> </td>  
  
</tr>  
  
<%  
    rs.MoveNext  
    Loop  
%>
```




Accessing a Database

- Opening a Table
 - Once a Recordset Object is created, its built-in *Open* method is available to extract information from a database table and make it available to our script for processing.
 - The general format for extracting ALL the information from a table is to open the entire table.

<%

...

```
RecordsetObject.Open "TableName",  
    ConnectionObject
```

...

TableName is the name of a database table.

Connection object is the name of an existing connection to the database containing the table.



ASP

MSTP

Accessing a Database

<%

...

```
Set Conn = Server.CreateObject("ADODB.Connection")
```

```
Set rs = Server.CreateObject("ADODB.Recordset")
```

```
dsn="DBQ=" & Server.MapPath("UsersDb/Users.mdb") &  
";Driver={Microsoft Access Driver (*.mdb)};"
```

```
Conn.Open dsn
```

```
rs.Open "Names", Conn
```

...

%>

With this statement in place, our script has extracted the entire Names table from the Users.mdb database and made it available for processing by the script.



Accessing a Database

- Selecting Records

- When the Names table is opened, the entire data can be loaded using the *Recordset* object.
- As in the Request.Form or Request.QueryString Collection, the data values contained in the recordset can be referenced through the notation rs ("FieldName"), where the *FieldName* is taken from the field names assigned when the table was created in Access.
- For instance, under FName and the LName fields in the Names' table you could be referenced them as rs ("FName") and rs ("LName").

<%=rs("FName")%>

<%=rs("LName")%>

We named the
recordset object
rs.



ASP

MSTP

Accessing a Database

```
<tr>
  <td width="63"><%=rs("Rank")%> </td>
  <td width="158"><%=rs("FName")%> </td>
  <td width="41"><%=rs("MI")%> </td>
  <td width="113"><%=rs("LName")%> </td>
  <td width="54"><%=rs("Unit")%> </td>
  <td width="152"><%=rs("Phone")%> </td>
</tr>
```

<u>Rank</u>	<u>First Name</u>	MI	<u>Last Name</u>	<u>Unit</u>	Phone
Capt	Miguel	A	<u>Ayala</u>	MSTP	7037846001



Accessing a Database

- Iterating thorough a Recordset
 - Databases have hundred, thousands of records.
 - We need to examine each record until we find a match or reach the End of File (EOF).
 - We can set up a program loop that advances the recordset cursor from one record to the next, looking for the matching criteria.
 - The loop will continue until we run out of records to check (the EOF property is true).
 - A solution: the **Do While...Loop**
 - It continues the loop until the Recordset object's EOF property is true.
 - We advance to the next record using the *MoveNext* method of the Recordset object.
 - *rs.MoveNext*

- Taking our previous script and modifying it:

```
<tr>
<td width="63"><%=rs("Rank")%> </td>
<td width="158"><%=rs("FName")%> </td>
<td width="41"><%=rs("MI")%> </td>
<td width="113"><%=rs("LName")%> </td>
<td width="54"><%=rs("Unit")%> </td>
<td width="152"><%=rs("Phone")%> </td>
</tr>
```

EOF → 04

Names : Table												
	ID	SSN	FName	LName	MI	Rank	Unit	Login	Password	Permissions	Phone	
▶	+	210	456-78-9123	Michael	Burke	J	LtCol	MSTP	burkemj	burkemj	admin	(703) 784-4972
	+	212	987-65-4321	Sean	Sadlier	M	Capt	MSTP	sadliersm	sadliersm	user	(703) 784-4315
	+	213	123-45-6789	Miguel	Ayala	A	Capt	MSTP	ayalama	ayalama	admin	(703) 784-6001
*	umber) - - 0											



Accessing a Database

- Checking for Matching Records
 - Done inside the Do...While loop.
 - Two ways to check for matching values:
 - Using If statements.
 - Using SQL statements.
 - For the login application, for example, we need to compare `rs("login")` with `Request ("login")` and `rs("Password")` with `Request ("Password")` for each record in the recordset, looking for matching values. If *both* matches are made, then we have found a valid login and password.

ASP



MSTP

Accessing a Database

- Using the *If* Statement

```
<%  
Dim password, login  
Password = Request("password")  
Login = Request("login")  
  
Dim dsn      'data source name & path    i.e. the database name and a path to it  
Dim Conn     'Connection object  
Dim rs       'recordset object  
  
Set Conn = Server.CreateObject("ADODB.Connection")  
Set rs = Server.CreateObject("ADODB.Recordset")  
  
dsn="DBQ=" & Server.MapPath("UsersDb/Users.mdb") & ";Driver={Microsoft Access Driver (*.mdb)}";  
  
Conn.Open dsn  
rs.Open "Names", Conn  
  
Do While Not rs.EOF  
    If rs("login") = Login AND rs("password") = Password Then  
        Response.Redirect "Show_Records.asp"  
    Else  
        Response.Redirect "Login.asp"  
    End If  
    Rs.MoveNext  
Loop  
...  
>%
```

The looping structure checks for a record in the database where the field login matches the variable Login obtained from the user through a form. In the same statement the same process is done for the password. Notice that **BOTH** criteria need to be

04/10/04



Accessing a Database

- Using SQL statement
 - Instead of looping through all the records in the database, we will let the SQL do the job for us.
 - You have to create a SQL statement to select the record that matches the Password and Login values respectively.
 - The general format is:

```
SELECT * FROM Names WHERE Login='the form Account value' AND  
Password='the form Password value'
```

- We want to SELECT all (*) fields (the Login field and the Password field) from the record in the Names table WHERE the value in the Login field of the table matches the Login value entered on the form, and the value in the Password field of the table matches the Password value entered on the form.



ASP

MSTP

Accessing a Database

- From the coding standpoint, we need to insert references inside the single quotes to the corresponding form values.
- In the case of our Login page:

```
"SELECT * FROM Names WHERE Login='"  
Request ("Login")  
' ' AND Password='"  
Request ("Password")  
' ' ' ' "
```

Information
coming from the
Form.

- Putting it all together:

```
"SELECT * FROM Names WHERE Login=' "& Request ("Login")& "' AND Password=' "&  
Request ("Password") & "' ' ' ' "
```

- We could assign it to a variable:

```
srtSQL = "SELECT * FROM Names WHERE Login=' "& Request ("Login")& "' AND  
Password=' "& Request ("Password") & "' ' ' ' "
```

04/10/04



ASP

MSTP

Accessing a Database

<%

```
Set Conn = Server.CreateObject("ADODB.Connection")  
Set rs = Server.CreateObject("ADODB.Recordset")
```

```
dsn="DBQ=" & Server.MapPath("UsersDb/Users.mdb") & ";Driver={Microsoft Access Driver (*.mdb)};"
```

```
Conn.Open dsn
```

```
strSQL = "SELECT * FROM Names WHERE Password = '" & Request("password") & "' and Login = '" &  
Request("login") & "'"
```

```
rs.Open strSQL, Conn
```

```
If RS.EOF = true Then
```

```
    Conn.Close 'closes the database connection
```

```
    set rs = Nothing 'sets the recordset equal to nothing
```

```
    set Conn= Nothing 'sets the connection object equal to nothing
```

```
    response.redirect "Login.asp"
```

```
End If
```

%>



Accessing a Database

- Closing Connections and Recordsets
 - Both Connection objects and Recordset objects have formal *Close* methods that can be applied when you are finished with either.
 - Standard programming practice normally dictates the you close open items when you are done with them.
 - Under ASP, however, this is not necessary. ASP automatically closes any open connections and recordsets when it finishes processing a page.

<%

...

```
Conn.Close 'closes the database connection
set rs = Nothing 'sets the recordset equal to nothing
set Conn= Nothing 'sets the connection object equal to
nothing
```

...

%> 04/10/04



Updating a Database

- Before you can make changes to a *Recordset*, you must make sure you do two things first:
 1. Open the *Recordset* object with the *adOpenStatic* or *adOpenDynamic* cursor used in the second argument of the *Recordset* object's *Open* method.
 2. Use *adLockOptimistic* for the lock type in the third argument of the *Recordset* object's *Open* method.

```
rs.Open strSQL, Conn,adOpenDynamic,adLockOptimistic,adCmdText
```

- After the above steps, you will be ready to edit the *Recordset*.

```
rs.Update
```




Updating a Database

- Adding records
 - The *AddNew* method. This method adds a new, blank record to the database.
 - Set the fields by assigning your data to the respective fields of the Recordset.
 - Execute the *Recordset.Update* method to commit all changes to the record.

```
rs.AddNew
```

```
rs("SSN")=Request("SSN")  
rs("FName") = Request("FName")  
rs("MI") = Request("MI")  
rs("LName") = Request("LName")  
rs("Rank") = Request("Rank")  
rs("Unit") = Request("Unit")  
rs("login") = Request("login")  
rs("Password") = Request("Password")  
rs("Permissions") = Request("Permissions")  
rs("Phone") = Request("Phone")
```




Updating a Database

- Updating records
 - First, you need to position the current pointer to the record that you wish to update. Use a proper SQL statement to achieve this (just like selecting a record for viewing).
 - Modify the record by assigning the new values to the fields that need changes only.
 - Finally, execute the *Update* method statement to write the changes back to the database.

`rs.Update`



Updating a Database

```
IDNum = Request("IDNum")
```

```
strSQL = "SELECT * FROM Names WHERE ID=" & IDNum
```

```
rs.Open strSQL, Conn, adOpenDynamic, adLockOptimistic
```

```
rs("SSN") = Request("SSN")
```

```
rs("FName") = Request("FName")
```

```
rs("MI") = Request("MI")
```

```
rs("LName") = Request("LName")
```

```
rs("Rank") = Request("Rank")
```

```
rs("Unit") = Request("Unit")
```

```
rs("Phone") = Request("Phone")
```

```
rs.Update
```

The SQL statement is selecting a record in which the ID number from the database matches the ID passed from the form. After selecting the record, the changes are made to the selected record.



Updating a Database

- Deleting Records

- There are two easy ways to delete a record.
 - Using the *Recordset.Delete* method
 - The *DELETE* statement in SQL.
- I will be discussing the *Recordset.Delete* method to be consistent using the *Recordset* object.
- Like in the update part, you need to position the current pointer to the record that you wish to update. Use a proper SQL statement as well.
- Delete the record by using the *Recordset.Delete* method

```
IDNum = Request("IDNum")
```

```
strSQL = "SELECT * FROM Names WHERE ID=" & IDNum  
rs.Open strSQL, Conn, adOpenDynamic, adLockOptimistic  
rs.Delete
```


ADVANCE TOPICS



MSTP

- How do I start developing my application?
- Object Oriented Programming
- Code Reuse



ADVANCE TOPICS

MSTP

How do I start developing my application?

- Analyze the problem
- What are the user needs?
- Who are the stakeholders?
- What are the requirements?
- Brainstorming
 - What information do I need?
 - Where is the information coming from?
 - What is the information flow?
- Storyboard





ADVANCE TOPICS

MSTP

How do I start developing my application?

- Analyze the problem
 - Problem definition
 - My understanding of the problem might not be the same as yours.
 - Who are the users?
 - Who will be affected by the application?
 - Who will be maintaining the application?
 - Documentation

04/10/04



ADVANCE TOPICS

MSTP

How do I start developing my application?

- Analyze the problem (cont)
 - Define the solution System Boundary
 - The system
 - Things that interact with the system
 - Where does it get the information?
 - What other external systems will interact with it?
 - Who will supply, use, or retrieve information from it?
 - Identify any constraints imposed in the solution
 - Operating System
 - Hardware



ADVANCE TOPICS

MSTP

How do I start developing my application?

- What do they really want?
 - They do not even know what they want in most cases.
 - What are the requirements of the system?
 - Inputs
 - Outputs
 - Users
 - Maintenance
 - Where is the information coming from?
 - What is the information flow?



ADVANCE TOPICS

MSTP

How do I start developing my application?

- Brainstorming
 - Start generating ideas on possible solutions.
 - Approach the problem from different angles to create more possible solutions
 - Once you have different COAs, analyze them and bring the best one out.



ADVANCE TOPICS

MSTP

How do I start developing my application?

- Storyboard
 - Provides an opportunity for user's reaction before efforts and time are committed to coding.
 - You walk the user through the processes of your solution.
 - Who the players are
 - What happen to them
 - How it happens

ADVANCE TOPICS

MSTP

Storyboard example

The user will be prompted for user name and password

File Reservations Customers Reports Help

File Reservations Customers Reports Help Logout

Login Query Reservations Payments Modify Reports

Welcome to the FARTS system

User name:

Password:

OK

Once logged in the user can query for a flight

File Reservations Customers Reports Help

File Reservations Customers Reports Help Logout

Query Login Reservations Payments Modify Reports

Query based on Customer Itinerary

Depart Location:

Arrival Location:

Depart Date:

Departure Time:

OK

Results of the query are displayed

File Reservations Customers Reports Help

File Reservations Customers Reports Help Logout

Query Login Reservations Payments Modify Reports

Results of Query

Flight#	Depart Time	Arrival Time	Fare	Airline Code	%Full

OK < Back

A booking for a flight session can be started

File Reservations Customers Reports Help

File Reservations Customers Reports Help Logout

Reservations Login Query Payments Modify Reports

Book Reservation

Agent Last: Agent First: ID

Credit card # Type: Exp:

Cust First: Cust Last:

Cust SSN: Phone #

Address: City:

State: Zip

Disabled: ☐ Chidren: ☐

Process Payment <<BACK



ADVANCE TOPICS

MSTP

Object Oriented Programming

- Object Oriented Programming
 - Provides encapsulation
 - Functionality is enclosed to such a degree that you are able to extract and use that functionality in a physically different environment
 - Separates implementation from interface
 - Modularity approach
 - Easy to build and test
 - Can be easily reuse



ADVANCE TOPICS

MSTP

Code Reuse

- Code Reuse
 - Saves on coding time
 - One of the most efficient ways of reducing development time and effort.
 - Less testing required
 - What makes reuse so important?
 - Encapsulation
 - Generic functionality



ASP

Putting it all together

Login.asp

Login/password fails

check_password
.asp

Login/password
successful

detail.as

Show_records.

search.a

Change_Info.a

Add_Kids.a

Add_user.asp

Delete_User.as

Results.as

Change_info_action.
asp

Add_kids_action.asp

Add_user_action.as
p

Delete_user_action.
asp

04/10/04



LAB 9: ALPHA ROSTER

MSTP

Final Product

- Create an alpha roster web application with the following functionalities:
 - **Login Page:** Ask the user for a logon id and a password. The page must be able to process the information either on its own or pass it to another .asp file to process the information. That pair needs to be check against information in a database. The end product is to be able to check the logon id and password provided by the user against a database.
 - **Database Connectivity:** The application must be able to have connectivity with a database where all the information about the users reside.
 - **Show Records:** Display the appropriate information to the user, retrieving it from the database.
 - **Update Records:** The application must be able to update existing information from the database.



LAB 9: ALPHA ROSTER

MSTP

Final Product (cont.)

- **Add Records:** The application must provide the functionality of adding new users to the database, collecting at least the following information:

1. SSN
2. Rank
3. First Name
4. MI
5. Last Name
6. Unit
7. Phone Number
8. Spouse Name
9. Kids Name
10. Kids Age
11. Permission for the database (User or Admin)
12. Password
13. Logon Id

- This functionality must be granted to an administrator

04/10/04



LAB 9: ALPHA ROSTER

MSTP

Final Product (cont.)

- **Delete Records:** The application must provide the functionality of deleting users. This functionality must be granted to an administrator.

REVIEW



MSTP

- During this class we discussed:
 - HTML Basics
 - Tags
 - Heading
 - Paragraph
 - Text formatting
 - Lists
 - Unordered
 - Ordered
 - Tables
 - Rows
 - Columns
 - Forms
 - IIS installation

REVIEW



MSTP

- Programming and Scripting in VBScript
 - Data type
 - Variant
 - Variables
 - Constants
 - Control Structures
 - If...Then...Else
 - Select Case
 - Looping Structures
 - Do...Loop
 - While...Wend
 - For...Next

REVIEW



MSTP

- ASP
 - Processing user input request (Request Object)
 - Session Object
 - Responding to the user (Response Object)
 - Accessing a Database
- Database Overview
 - Fields
 - Records
 - Tables
 - Building a Database with MS Access

REVIEW



MSTP

- Introduction to Structured Query Language (SQL)
 - Select statement
 - The Where clause
 - Order By clause
- Advance Topics
 - How do I start developing my application?
 - Analyze the problem
 - What are the user needs?
 - Who are the stakeholders?
 - What are the requirements?
 - Brainstorming
 - Storyboard
 - Object Oriented Programming
 - Code Reuse

QUESTIONS



MSTP



04/10/04

137



POINTS OF CONTACT

MSTP

LtCol Michael J. Burke
(703) 784-4972 DSN 278
burkemj@mstp.quantico.usmc.mil

Capt Miguel A. Ayala
(703) 784-6001 DSN 278
ayalama@mstp.quantico.usmc.mil

"Training The First To Fight"

www.mstp.usmc.mil